ELSEVIER

Journal of Computational and Applied Mathematics 71 (1996) 267–286

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

Adaptive Richardson iteration based on Leja points

D. Calvetti [a,1], L. Reichel [b,*]

[a] *Department of Pure and Applied Mathematics, Stevens Institute of Technology, Hoboken, NJ 07030, USA*

[b] *Department of Mathematics and Computer Science, Kent State University, Kent, OH 44242, USA*

Received 2 February 1995; revised 15 October 1995

Abstract

An adaptive Richardson iteration method is presented for the solution of large linear systems of equations with a sparse symmetric positive definite matrix. The relaxation parameters for Richardson iteration are chosen to be reciprocal values of Leja points for an interval $[a,b]$ on the positive real axis, and the endpoints a and b are determined adaptively by computing certain modified moments during the iterations. Computed examples show that the adaptive Richardson method can be competitive with the conjugate gradient algorithm.

Keywords: Iterative method; Linear system; Modified moments

AMS classification: 65F10; 42C05

1. Introduction

The solution of large linear systems of equations

$$Ax = b, \quad A \in \mathbb{R}^{n\times n}, \quad x, b \in \mathbb{R}^n, \tag{1.1}$$

where the matrix A is sparse, symmetric and positive definite arises in many scientific applications. Several attractive iterative methods for the solution of (1.1) are available. These include the conjugate gradient method, Chebyshev iteration and Richardson iteration; see [8, 9, 11, 17]. It is fairly easy to achieve efficient implementations of the Richardson and Chebyshev iteration methods on vector and parallel computers, and this has generated renewed interest in these methods; see [2, 15]. When applying Chebyshev and Richardson iteration one typically requires that an interval $[a,b]$, $0 < a \leqslant b < \infty$, that contains the spectrum of the matrix A be explicitly known. Golub and

* Corresponding author. e-mail: na.reichel@na-net.ornl.gov. Research supported in part by NSF grants DMS-9002884 and DMS-9205531.

[1] Research supported in part by the Design and Manufacturing Institute at Stevens Institute of Technology. e-mail: na.calvetti@na-net.ornl.gov.

SSDI 0377-0427(95)000231-6

Kent [7] proposed an adaptive Chebyshev iteration method, in which approximations of such an interval are determined by computing certain modified moments during the iterations. Hageman and Young [11] and Jea and Young [12] describe an adaptive Chebyshev iterative method in which they seek to determine a good interval $[a,b]$ by monitoring the rate of decrease of the residual error.

The present paper describes how modified moments can be applied to determine approximations of an interval containing the spectrum of A when carrying out Richardson iteration. Let $[a_j,b_j]$ denote a computed approximation of the smallest interval containing the spectrum of A, and assume that this interval is updated during the iterations to yield the new interval $[a_{j+1},b_{j+1}]$. We choose the relaxation parameters for Richardson iteration after this update as reciprocal values of Leja points (defined below) for $[a_{j+1},b_{j+1}]$ in a manner that takes the distribution of previously used relaxation parameters into account. Leja points associated with previously chosen parameters serve as "memory" of previous iterations when new relaxation parameters are selected. The presence of this memory makes our iterative scheme adapt quickly when the interval $[a_j,b_j]$ is updated. Numerical examples show that our adaptive Richardson iterative method can be competitive with the conjugate gradient method.

Let $\boldsymbol{x}_0$ be a given initial approximate solution to (1.1). We can write Richardson iteration in the form

$$\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \delta_k \boldsymbol{r}_k, \quad k = 0,1,\dots ,$$

where

$$\boldsymbol{r}_k := \boldsymbol{b} - A\boldsymbol{x}_k$$

is the residual vector associated with the approximate solution $\boldsymbol{x}_k$ and $\delta_k \in \mathbb{R}$ is a relaxation parameter. We wish to determine the relaxation parameters so that the errors

$$\boldsymbol{e}_k := A^{-1}\boldsymbol{b} - \boldsymbol{x}_k, \quad k = 0,1,\dots, \tag{1.2}$$

converge to zero rapidly as k increases. The residual vectors can be expressed as

$$\boldsymbol{r}_k = p_k(A)\boldsymbol{r}_0, \quad k = 0,1,\dots, \tag{1.3}$$

where the polynomials p_k satisfy the recurrence relation

$$p_{k+1}(\lambda) = (1 - \delta_k\lambda)p_k(\lambda), \quad k = 0,1,\dots, \tag{1.4}$$

with $p_0(\lambda) := 1$. Because of relation (1.3), we refer to the p_k as residual polynomials.

Edrei [3] and Leja [13] introduced recursively defined points associated with compact sets in the complex plane known as Leja points. Our adaptive Richardson scheme chooses the relaxation parameters δ_k to be reciprocal values of Leja points associated with intervals $[a_j,b_j]$ on the positive real axis. We want these intervals to contain (most of) the spectrum of A, and they are determined during the iterations in the following manner: we compute certain modified moments during the iterations and use the modified moments and relaxation parameters as input to the modified Chebyshev algorithm. This algorithm computes recursion coefficients of orthogonal polynomials from modified moments; see [5] for a discussion of its properties. The recursion coefficients obtained from $2m$ modified moments determine an $m \times m$ symmetric tridiagonal matrix $\hat{T}_m$, whose

eigenvalues approximate some eigenvalues of A. Associated with $2m$ modified moments is an m-point Gaussian quadrature rule. The nodes of this rule are given by the eigenvalues of $\hat{T}_m$ and the weights are given by the square of the first components of the normalized eigenvectors of $\hat{T}_m$. The extreme eigenvalues of $\hat{T}_m$, whose associated Gaussian weights are not "tiny", are used to update the interval $[a_j, b_j]$. Eigenvalues of $\hat{T}_m$ associated with tiny weights are ignored, because these eigenvalues are highly sensitive to perturbations in the modified moments, e.g., caused by round-off errors.

Section 2 discusses the convergence of our iterative scheme under the assumption that the interval $[a_j, b_j]$ contains the spectrum of A. The computation of modified moments and application of the modified Chebyshev algorithm are discussed in Section 3. Section 4 discusses the significance of the Gaussian weights. Our adaptive Richardson scheme is presented in Section 5. Computed examples can be found in Section 6, and Section 7 contains concluding remarks.

2. Leja points and Richardson iteration

Edrei [3] and Leja [13] introduced a sequence of points known as Leja points for fairly general compact sets in the complex plane $\mathbb{C}$. The application of Leja points to Richardson iteration when the spectrum of A lies in a compact set $\mathbb{K}$ in $\mathbb{C}\backslash\{0\}$ is discussed in [4, 14]. This section reviews these results for the special case when $\mathbb{K} = [a, b]$, $0 < a < b < \infty$.

Introduce the weight function $\omega(\lambda) = \lambda$. Let

$$z_0 := b \tag{2.1}$$

and choose z_k, so that

$$\prod_{j=0}^{k-1} |z_k - z_j|\omega(z_k) = \max_{z\in\mathbb{K}} \prod_{j=0}^{k-1} |z - z_j|\omega(z), \quad z_k \in \mathbb{K}, \quad k = 1, 2, \ldots. \tag{2.2}$$

Note that (2.2) might not determine the points z_k uniquely. We call any points $z_0, z_1, \ldots$ which satisfy (2.1)–(2.2) Leja points for $\mathbb{K}$. Edrei [3] and Leja [13] studied these points when the weight function in (2.2) is $\omega(\lambda) = 1$. The asymptotic properties of the Leja points are the same for the weight functions $\omega(\lambda) = \lambda$ and $\omega(\lambda) = 1$, but computed examples in [14] show the former weight function to yield a better choice of the first few relaxation parameters $\delta_k := 1/z_k$.

It follows from the results in [13] that the Leja points for $[a, b]$ are uniformly distributed on $[a, b]$ with respect to the density function

$$\mathrm{d}\sigma(\lambda) := \frac{1}{\pi}(\lambda - a)^{-1/2}(b - \lambda)^{-1/2}. \tag{2.3}$$

We note in passing that the zeros of Chebyshev polynomials for the interval $[a, b]$,

$$T_\ell^{(a,b)}(t) := \cos\left(\ell \arccos\left(\frac{b-a}{2}t + \frac{b+a}{2}\right)\right), \quad \ell = 0, 1, \ldots, \tag{2.4}$$

also are uniformly distributed on $[a, b]$ with respect to the density function (2.3) as $\ell \to \infty$.

We turn to the convergence of the Richardson method with parameters $\delta_k := 1/z_k$, where the z_k are Leja points for $\mathbb{K} = [a,b]$. Let $\boldsymbol{\delta} := \{\delta_k\}_{k=0}^{\infty}$ and introduce the asymptotic convergence factor

$$\kappa(\boldsymbol{\delta}) := \overline{\lim_{k\to\infty}} \sup_{\boldsymbol{e}_0\neq\boldsymbol{0}} \left(\frac{\|\boldsymbol{e}_k\|}{\|\boldsymbol{e}_0\|}\right)^{1/k}, \tag{2.5}$$

where $\|\cdot\|$ denotes the Euclidean norm and the error vectors $\boldsymbol{e}_k$ are given by (1.2). Introduce the spectral resolution

$$A = Q\Lambda Q^{\mathrm{T}}, \tag{2.6}$$

where

$$\Lambda = \operatorname{diag}[\lambda_1, \lambda_2, \ldots, \lambda_n], \qquad \lambda_1 \leqslant \lambda_2 \leqslant \cdots \leqslant \lambda_n,$$

$$Q = [\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_n], \quad Q^{\mathrm{T}}Q = I.$$

Substituting

$$\boldsymbol{e}_k = Q p_k(\Lambda) Q^{\mathrm{T}} \boldsymbol{e}_0$$

into (2.5) yields

$$\kappa(\boldsymbol{\delta}) = \overline{\lim_{k\to\infty}} \max_{\lambda\in\lambda(A)} |p_k(\lambda)|^{1/k},$$

where $\lambda(A)$ denotes the spectrum of A. In general, $\lambda(A)$ is not explicitly known. We therefore assume that an interval $\mathbb{K} := [a,b]$ which contains $\lambda(A)$ is explicitly known, and introduce the asymptotic convergence factor with respect to $\mathbb{K}$,

$$\kappa(\boldsymbol{\delta}, \mathbb{K}) := \overline{\lim_{k\to\infty}} \max_{\lambda\in\mathbb{K}} |p_k(\lambda)|^{1/k} \geqslant \kappa(\boldsymbol{\delta}).$$

We would like to choose the iteration parameters δ_k so that $\kappa(\boldsymbol{\delta}, \mathbb{K})$ is minimized.

Lemma 2.1 (Convergence). *Let $\lambda(A) \subset \mathbb{K} = [a,b]$ with $0 < a < b < \infty$. Assume that the iteration parameters δ_j are reciprocal values of Leja points for $\mathbb{K}$ and let $\boldsymbol{\delta} := \{\delta_k\}_{k=0}^{\infty}$. Then*

$$\kappa(\boldsymbol{\delta}, \mathbb{K}) = \inf_{\boldsymbol{\eta}} \kappa(\boldsymbol{\eta}, \mathbb{K}) = \rho(a,b) := \frac{\mu}{1+\sqrt{1-\mu^2}}, \tag{2.7}$$

where $\mu = (b-a)/(b+a)$. In particular, if $\mathbb{K}'$ is a subinterval of $\mathbb{K}$, then

$$\inf_{\boldsymbol{\eta}} \kappa(\boldsymbol{\eta}, \mathbb{K}') \leqslant \inf_{\boldsymbol{\eta}} \kappa(\boldsymbol{\eta}, \mathbb{K}). \tag{2.8}$$

Proof. The Chebyshev polynomials for the interval $[-1,1]$ can be written as $T_j(\lambda) = \cosh(j \cosh^{-1}(\lambda))$ for $|\lambda| \geqslant 1$. Let p_j be polynomials that satisfy (1.4). It is well known that

$$\min_{\boldsymbol{\delta}} \max_{\lambda\in\mathbb{K}} |p_j(\lambda)| = \frac{1}{|T_j(\frac{1}{\mu})|}, \tag{2.9}$$

see, e.g., [11, Section 4.2]. The equality on the right-hand side of (2.7) follows from (2.9) and properties of the Chebyshev polynomials; see [9, 11, 17] for details. A proof of the left-hand side

equality of (2.7) can be found in [4, 14] and follows from the result of Leja [13] that Leja points for $\mathbb{K}$ are uniformly distributed with respect to the density function (2.3). Inequality (2.8) follows from the observation that $\partial\rho/\partial a < 0$ and $\partial\rho/\partial b > 0$. □

The smallest interval allowed in Lemma 2.1 is $[a,b] = [\lambda_1, \lambda_n]$. In view of (2.8), we call this interval the asymptotically optimal interval for A.

3. Modified moments

In this section we define modified moments and describe how they can be used to determine estimates of the extreme eigenvalues of A while computing the iterates $\boldsymbol{x}_k$ by Richardson iteration. Our formulas are analogous with those derived by Golub and Kent [7] for computing eigenvalue estimates while computing iterates by the Chebyshev iteration method. Let A have spectral resolution (2.6) and let p_k be the residual polynomials (1.3). Express $\boldsymbol{r}_0$ in the basis of eigenvectors

$$\boldsymbol{r}_0 = \sum_{j=1}^{n} w_j \boldsymbol{q}_j. \tag{3.1}$$

Then it follows from $A = Q\Lambda Q^{\mathrm{T}}$ that

$$\boldsymbol{r}_k = \sum_{j=1}^{n} w_j p_k(\lambda_j)\boldsymbol{q}_j.$$

Introduce the density function

$$\mathrm{d}\sigma_n := \sum_{j=1}^{n} w_j^2 \delta(\lambda - \lambda_j), \tag{3.2}$$

where the λ_j are the eigenvalues of A, the w_j are the coefficients defined by (3.1) and $\delta(\lambda)$ is the Dirac δ-function. We refer to $\mathrm{d}\sigma_n$ as the spectral density function for A associated with $\boldsymbol{r}_0$. Define the inner product

$$(\boldsymbol{r}_k, \boldsymbol{r}_\ell) := \boldsymbol{r}_k^{\mathrm{T}}\boldsymbol{r}_\ell = \sum_{j=1}^{n} w_j^2 p_k(\lambda_j)p_\ell(\lambda_j) = \int_{-\infty}^{\infty} p_k(\lambda)p_\ell(\lambda)\,\mathrm{d}\sigma_n.$$

From the vectors $\boldsymbol{r}_k$ and $\boldsymbol{r}_0$ one can compute the modified moment for $\mathrm{d}\sigma_n$ with respect to the residual polynomial p_k by

$$v_k := \int_{-\infty}^{\infty} p_k(\lambda)\,\mathrm{d}\sigma_n = (\boldsymbol{r}_k, \boldsymbol{r}_0) \tag{3.3}$$

without explicitly knowing $\mathrm{d}\sigma_n$. Assume that we have computed the $2m$ modified moments $\{v_k\}_{k=0}^{2m-1}$. We now can apply the modified Chebyshev algorithm to determine recursion coefficients α_k and β_k for the first m monic orthogonal polynomials with respect to $\mathrm{d}\sigma_n$ from the modified moments $\{v_k\}_{k=0}^{2m-1}$ and the relaxation parameters $\{\delta_k\}_{k=0}^{2m-1}$, which also are recursion coefficients for the residual polynomials; see (1.4). The modified Chebyshev algorithm is analyzed by Gautschi [5].

Algorithm 3.1 (Modified Chebyshev algorithm)
Input: m, $\{\nu_k\}_{k=0}^{2m-1}$, $\{\delta_k\}_{k=0}^{2m-1}$;
Output: $\{\alpha_k\}_{k=0}^{m-1}$, $\{\beta_k\}_{k=0}^{m-1}$;
for $j := 0, 1, \ldots, 2m-1$ **do**
$\sigma_{-1,j} := 0$; $\sigma_{0,j} := \nu_j$;
end j;
$\alpha_0 := \dfrac{1}{\delta_0} - \dfrac{\nu_1}{\nu_0}\dfrac{1}{\delta_0}$; $\beta_0 := 0$;
for $k := 1, 2, \ldots, m-1$ **do**
for $j := k, k+1, \ldots, 2m-k-1$ **do**

$$\sigma_{k,j} := -\frac{1}{\delta_j}\sigma_{k-1,j+1} + \left(\frac{1}{\delta_j} - \alpha_{k-1}\right)\sigma_{k-1,j} - \beta_{k-1}\sigma_{k-2,j};$$

end j;

$$\alpha_k := \frac{1}{\delta_k} - \frac{1}{\delta_k}\frac{\sigma_{k,k+1}}{\sigma_{k,k}} + \frac{1}{\delta_{k-1}}\frac{\sigma_{k-1,k}}{\sigma_{k-1,k-1}}; \qquad \beta_k := -\frac{1}{\delta_{k-1}}\frac{\sigma_{k,k}}{\sigma_{k-1,k-1}};$$

end k;

We assume that m in Algorithm 3.1 is chosen small enough so that $\sigma_{k,k} \neq 0$ for $0 \leqslant k < m$. The recursion coefficients α_k and β_k computed by the algorithm determine the tridiagonal matrix

$$T_m := \begin{bmatrix} \alpha_0 & 1 & & & 0 \\ \beta_1 & \alpha_1 & 1 & & \\ & \beta_2 & \alpha_2 & 1 & \\ & & & \ddots & \\ & & \ddots & \ddots & 1 \\ 0 & & & \beta_{m-1} & \alpha_{m-1} \end{bmatrix}. \tag{3.4}$$

A diagonal similarity transformation by the matrix

$$D_m := \operatorname{diag}\left[1, \beta_1^{1/2}, \ldots, \prod_{k=1}^{m-1} \beta_k^{1/2}\right]$$

yields the symmetric tridiagonal matrix

$$\hat{T}_m := D_m^{-1} T_m D_m. \tag{3.5}$$

The matrix $\hat{T}_m$ can also be computed by applying the Lanczos process to the matrix A with initial vector $\boldsymbol{r}_0$; see, e.g., [8, Ch. 9] for a discussion of the Lanczos process. Therefore

$$\lambda_1 \leqslant \hat{\lambda}_1, \qquad \hat{\lambda}_m \leqslant \lambda_n, \tag{3.6}$$

where $\hat{\lambda}_1 \leqslant \hat{\lambda}_2 \leqslant \cdots \leqslant \hat{\lambda}_m$ are the eigenvalues of $\hat{T}_m$.

4. Gaussian weights

The eigenvalues $\{\hat{\lambda}_j\}_{j=1}^m$ of $\hat{T}_m$ are the nodes and the square of the first components of the eigenvectors of length $v_0^{1/2}$ are the weights $\hat{w}_j^2$ of an m-point Gaussian quadrature rule associated with the density function (3.2),

$$G_m f := \sum_{j=1}^{m} f(\hat{\lambda}_j)\hat{w}_j^2. \tag{4.1}$$

The nodes and weights of the quadrature rule (4.1) can conveniently be computed from $\hat{T}_m$ by the Golub–Welsch [10] algorithm. We note that if $m = n$, then $\hat{\lambda}_j = \lambda_j$ and $\hat{w}_j^2 = w_j^2$ for $1 \leqslant j \leqslant n$. The density function

$$\mathrm{d}\hat{\sigma}_m(\lambda) := \sum_{j=1}^{m} \delta(\lambda - \hat{\lambda}_j)\hat{w}_j^2 \tag{4.2}$$

associated with (4.1) provides an approximation of $\mathrm{d}\sigma_n$ in the following sense:

(i) the first $2m$ moments associated with $\mathrm{d}\sigma_m$ and $\mathrm{d}\sigma_n$ are the same;

(ii) between each pair of adjacent Gaussian nodes $\{\hat{\lambda}_j, \hat{\lambda}_{j+1}\}$, there is an eigenvalue of A, see [6];

(iii) the distribution function σ_n grows rapidly between three adjacent nodes if $\hat{\sigma}_m$ does, where $\hat{\sigma}_m$ denotes the cumulative distribution function associated with $\mathrm{d}\hat{\sigma}_m$.

Property (iii) is stated in a precise manner by the Separation Theorem, shown independently by Chebyshev, Markoff and Stieltjes.

Lemma 4.1 (Separation Theorem [16, p. 50]). *Let the nodes $\hat{\lambda}_j$ be arranged in increasing order. Then there are numbers $\eta_j \in \mathbb{R}$, $a < \eta_1 < \eta_2 < \cdots < \eta_{m-1} < b$, such that*

$$\hat{w}_j^2 = \sigma_n(\eta_j) - \sigma_n(\eta_{j-1}), \quad 1 \leqslant j \leqslant m,$$

and

$$\hat{\lambda}_j < \eta_j < \hat{\lambda}_{j+1}, \quad 1 \leqslant j < m,$$

where $\eta_0 = \lambda_1$ and $\eta_m = \lambda_n$.

Three proofs of this lemma and further details can be found in [16]. In order to determine the quantities η_j of the lemma it may be necessary to modify σ_n at some of its jump discontinuities.

It follows from Lemma 4.1 that $\hat{w}_j^2 \leqslant \sigma_n(\hat{\lambda}_{j+1}) - \sigma_n(\hat{\lambda}_{j-1})$, and, therefore, a large Gaussian weight $\hat{w}_j^2$ implies that σ_n grows rapidly in the interval $I_j = [\hat{\lambda}_{j-1}, \hat{\lambda}_{j+1}]$. The growth of σ_n may depend on the fact that I_j contains many eigenvalues of A, or that $\boldsymbol{r}_0$ contains large components of eigenvectors associated with eigenvalues in I_j.

Conversely, a small Gaussian weight implies that the associated Gaussian node is very sensitive to perturbations in the modified moments. This statement is made precise below. We remark that our discussion is independent of the scaling of the residual vector $\boldsymbol{r}_0$, and therefore it is convenient to assume that $v_0 = 1$. The weights then satisfy

$$\sum_{j=1}^{m} \hat{w}_j^2 = 1. \tag{4.3}$$

Lemma 4.2 (Perturbation). *Let* $\mathrm{d}y$ *denote the differential of* y *and let, as usual, the* δ_j *be relaxation parameters. Then*

$$\mathrm{d}\hat{\lambda}_j = \frac{1}{\hat{w}_j^2} \sum_{k=0}^{2m-1} \mathrm{d}v_k \prod_{l=0}^{k-1} \left(-\frac{1}{\delta_l}\right) \left[\frac{1}{\delta_0}, \frac{1}{\delta_1}, \ldots, \frac{1}{\delta_k}\right] C_j, \quad 1 \leqslant j \leqslant m, \tag{4.4}$$

where

$$C_j(\lambda) := (\lambda - \hat{\lambda}_j) \prod_{\substack{k=1 \\ k \neq j}}^{m} \left(\frac{\lambda - \hat{\lambda}_k}{\hat{\lambda}_j - \hat{\lambda}_k}\right)^2, \tag{4.5}$$

and $[z_0, z_1, \ldots, z_k]C_j$ *denotes the* k*th divided differences of* C_j *defined by the points* $z_0, z_1, \ldots, z_k$.

Proof. Our proof is similar to the analysis by Gautschi [5] of the condition number of the map F defined below. Essentially, the lemma can also be found in [1]. We note that

$$\sum_{j=1}^{m} p_k(\hat{\lambda}_j)\hat{w}_j^2 = v_k, \quad 0 \leqslant k < 2m,$$

and introduce the map $F: \mathbb{R}^{2m} \to \mathbb{R}^{2m}$; $\{\hat{w}_1^2, \ldots, \hat{w}_m^2, \hat{\lambda}_1, \ldots, \hat{\lambda}_m\} \to \{v_k\}_{k=0}^{2m-1}$. The Jacobian of F is given by $J := PW$, where

$$P := \begin{bmatrix} p_0(\hat{\lambda}_1) & \cdots & p_0(\hat{\lambda}_m) & p_0'(\hat{\lambda}_1) & \cdots & p_0'(\hat{\lambda}_m) \\ p_1(\hat{\lambda}_1) & \cdots & p_1(\hat{\lambda}_m) & p_1'(\hat{\lambda}_1) & \cdots & p_1'(\hat{\lambda}_m) \\ \vdots & & \vdots & \vdots & & \vdots \\ p_{2m-1}(\hat{\lambda}_1) & \cdots & p_{2m-1}(\hat{\lambda}_m) & p_{2m-1}'(\hat{\lambda}_1) & \cdots & p_{2m-1}'(\hat{\lambda}_m) \end{bmatrix}$$

and

$$W := \mathrm{diag}[1, 1, \ldots, 1, \hat{w}_1^2, \hat{w}_2^2, \ldots, \hat{w}_m^2].$$

The Jacobian of F^{-1} is given by $J^{-1} := W^{-1}P^{-1}$. Therefore

$$\begin{bmatrix} \mathrm{d}\hat{w}_1^2 \\ \vdots \\ \mathrm{d}\hat{w}_m^2 \\ \mathrm{d}\hat{\lambda}_1 \\ \vdots \\ \mathrm{d}\hat{\lambda}_m \end{bmatrix} = W^{-1}P^{-1} \begin{bmatrix} \mathrm{d}v_0 \\ \mathrm{d}v_1 \\ \\ \vdots \\ \\ \mathrm{d}v_{2m-1} \end{bmatrix}. \tag{4.6}$$

We write the inverse of P in the form

$$P^{-1} = \begin{bmatrix} A \\ B \end{bmatrix},$$

where A and B are $m \times 2m$ matrices. The elements b_{jk} of B satisfy

$$\sum_{k=0}^{2m-1} b_{jk}\, p_k(\hat{\lambda}_l) = 0, \quad 1 \leqslant l \leqslant m,$$

$$\sum_{k=0}^{2m-1} b_{jk}\, p_k'(\hat{\lambda}_l) = \delta_{jl}, \quad 1 \leqslant l \leqslant m,$$

where δ_{jk} is the Kronecker δ-function. For each j such that $1 \leqslant j \leqslant m$, the polynomial (4.5) satisfies

$$C_j(\hat{\lambda}_l) = 0, \qquad C_j'(\hat{\lambda}_l) = \delta_{jl}, \quad 1 \leqslant l \leqslant m,$$

and is of degree $2m - 1$. Therefore,

$$C_j(\lambda) = \sum_{k=0}^{2m-1} b_{jk}\, p_k(\lambda), \quad 1 \leqslant j \leqslant m. \tag{4.7}$$

Explicit formulas for the coefficients b_{jk} can be determined by evaluation of the right- and left-hand sides of (4.7) at $\lambda = 1/\delta_j$, $j = 0, 1, \ldots$. This completes the proof of the theorem. □

Eq. (4.4) shows that $\mathrm{d}\hat{\lambda}_j$ is proportional to $1/\hat{w}_j^2$. This suggests that nodes $\hat{\lambda}_j$ associated with tiny weights $\hat{w}_j^2$ are sensitive to perturbations in the modified moments. We therefore ignore eigenvalue estimates $\hat{\lambda}_j$ associated with small Gaussian weights $\hat{w}_j^2$ in our scheme for updating the interval $[a, b]$ used to determine relaxation parameters for Richardson iteration. Details of this scheme are presented in the next section.

5. An adaptive Richardson iteration method

This section describes our adaptive Richardson iteration scheme and discusses some computational aspects. We first describe how the relaxation parameters are computed.

5.1. Determination of relaxation parameters

We want the initial interval $[a_0, b_0]$ to be a subset of the interval $[\lambda_1, \lambda_n]$. For instance, we may choose $a_0 := b_0 := (1/n)\,\mathrm{trace}(A)$. Assume that we know an interval $[a_j, b_j]$ such that $[a_j, b_j] \subset [\lambda_1, \lambda_n]$, and assume further that we have carried out q iterations by the Richardson method with relaxation parameters $\delta_0, \delta_1, \ldots, \delta_{q-1}$. We then would like to determine new relaxation parameters δ_k, $k \geqslant q$, as reciprocal values of Leja points for the set $\mathbb{K} = [a_j, b_j]$ in the presence of the points $z_k = 1/\delta_k$ for $0 \leqslant k < q$. However, the numerical determination of a large number of Leja points according to formulas (2.1) and (2.2) may be quite cumbersome. We therefore discretize the interval $[a_j, b_j]$ using grid points $\{t_i\}_{i=1}^{\ell}$, which we choose to be the zeros of a Chebyshev polynomial $T_\ell^{(a_j, b_j)}$ for the interval $[a_j, b_j]$, defined by (2.4). The degree ℓ is chosen sufficiently large to make the discretization error negligible. In view of that

$$\prod_{j=0}^{k-1} (\lambda - z_j) = p_k(\lambda) \prod_{j=0}^{k-1} (-z_j),$$

Leja points for $\mathbb{K} = [a_j, b_j]$ can be determined by maximizing $|p_k(\lambda)|$ over $\mathbb{K}$. This maximum is used in our criterion for when to update the interval $\mathbb{K}$, see below, and is part of the output of the following algorithm.

Algorithm 5.1 (Computation of relaxation parameters)
Input: a_j, b_j, ℓ, q, $\{z_k\}_{k=0}^{q-1}$ $(z_k = 1/\delta_k)$;
Output: δ_q, $\mu_q := \max_{a_j \leqslant \lambda \leqslant b_j} |p_q(\lambda)|$;
Let $\mathbb{K}_\ell := \{t_i\}_{i=1}^{\ell}$ *be the set of zeros of a Chebyshev polynomial of the first kind of degree* ℓ *for the interval* $[a_j, b_j]$;
if $q = 0$ **then**
$z_0 := b_j$; $\delta_0 := 1/z_0$; $\mu_0 := 1$;
else
Determine $z_q \in \mathbb{K}_\ell$, *so that*

$$|p_q(z_q)|\omega(z_q) = \max_{z \in \mathbb{K}_\ell} |p_q(z)|\omega(z);$$

$\delta_q := 1/z_q$; $\mu_q := |p_q(z_q)|$;
endif ;

The points $z_0, z_1, \ldots, z_{q-1}$ serve as memory of previous iterations when the new relaxation parameter δ_q is determined. The presence of this memory has the effect that relaxation parameters δ_q determined after the interval $\mathbb{K} = [a_j, b_j]$ has been increased, are distributed so that eigenvector components in the residual error that have not been damped before will be damped more heavily than other eigenvector components until the points $z_0, z_1, \ldots, z_q$ for some $q \geqslant k$ are distributed roughly according to the density function (2.3).

5.2. Updating the interval

In Section 3 we showed how to compute modified moments associated with the residual polynomials p_k during the iterations, and how these modified moments and the relaxation parameters can be used to determine a symmetric tridiagonal matrix $\hat{T}_m$. Assume for the moment that the eigenvalues $\hat{\lambda}_1 \leqslant \hat{\lambda}_2 \leqslant \cdots \leqslant \hat{\lambda}_m$ and Gaussian weights $\hat{w}_j^2$ associated with $\hat{T}_m$ have been computed. We will now discuss their application to updating the interval $[a_j, b_j]$. Thus, we would like to determine a new interval $[a_{j+1}, b_{j+1}]$ that gives faster convergence than $[a_j, b_j]$ and satisfies

$$0 < a_{j+1} \leqslant a_j, \quad b_j \leqslant b_{j+1}.$$

An obvious way to define the new interval $[a_{j+1}, b_{j+1}]$ is given by

$$a_{j+1} := \min\{a_j, \hat{\lambda}_1\}, \tag{5.1}$$

$$b_{j+1} := \max\{b_j, \hat{\lambda}_m\}. \tag{5.2}$$

However, computational experience from the solution of numerous problems indicates that when the Gaussian weight $\hat{w}_1^2$ associated with $\hat{\lambda}_1$ is tiny, the computed value of $\hat{\lambda}_1$ may be contaminated by a large error and should not be used to update the interval; cf. Lemma 4.2. Similarly, when $\hat{w}_m^2$ is

tiny, the computed value of $\hat{\lambda}_m$ should not be used to update the interval. Therefore, we replace the updating formulas (5.1) and (5.2) in our adaptive Richardson method by

$$\textbf{if } \hat{w}_1^2 \geqslant \varepsilon_w \textbf{ then } a_{j+1} := \min\{a_j, \hat{\lambda}_1\} \textbf{ else } a_{j+1} := a_j, \tag{5.3}$$

$$\textbf{if } \hat{w}_m^2 \geqslant \varepsilon_w \textbf{ then } b_{j+1} := \max\{b_j, \hat{\lambda}_m\} \textbf{ else } b_{j+1} := b_j, \tag{5.4}$$

where we assume that the weights have been normalized to satisfy (4.3). The choice of ε_w is not very critical. We have found that, when the rate of convergence can be increased significantly by letting a_{j+1} be smaller than a_j, the Gaussian weight $\hat{w}_1^2$ typically is quite large. Similarly, when the rate of convergence can be increased significantly by letting b_{j+1} be larger than b_j, the weight $\hat{w}_m^2$ typically is large. In the computed examples we used $\varepsilon_w = 0.01/m$. Our computational experience suggests that for reason of numerical stability m should be chosen fairly small, e.g., $m = 5$.

5.3. Other sets of modified moments

Formula (3.3) shows how to compute one modified moment in each iteration. The application of this formula would require $2m - 1$ iterations, i.e., the computation of $\boldsymbol{r}_0, \boldsymbol{r}_1, \ldots, \boldsymbol{r}_{2m-1}$, in order to determine the matrix $\hat{T}_m$. We now show how to generate two modified moments in each iteration. This makes it possible to update the intervals $[a_j, b_j]$ more frequently. The ability to update the interval after only a few iterations is particularly important in the beginning of the iterations when many of the eigenvalues of A might lie far outside the initial interval $[a_0, b_0]$ used to determine the first relaxation parameters.

Consider the polynomials

$$\tilde{p}_{2k-1}(\lambda) := (1 - \delta_{k-1}\lambda)[p_{k-1}(\lambda)]^2 = p_{k-1}(\lambda)p_k(\lambda),$$
$$\tilde{p}_{2k}(\lambda) := [p_k(\lambda)]^2.$$

They satisfy the recursion relation

$$\tilde{p}_{k+1}(\lambda) = (1 - \tilde{\delta}_k\lambda)\tilde{p}_k(\lambda), \quad k = 0, 1, \ldots,$$

where

$$\tilde{\delta}_{2k+1} := \tilde{\delta}_{2k} := \delta_k, \quad k = 0, 1, \ldots. \tag{5.5}$$

It follows that the modified moments

$$\tilde{v}_k := \int_{-\infty}^{\infty} \tilde{p}_k(\lambda)\, \mathrm{d}\sigma_n$$

associated with the polynomials $\tilde{p}_k$ can be computed from the residual vectors $\boldsymbol{r}_{k-1}$ and $\boldsymbol{r}_k$ according to

$$\tilde{v}_{2k-1} = (\boldsymbol{r}_{k-1}, \boldsymbol{r}_k), \tag{5.6}$$

$$\tilde{v}_{2k} = (\boldsymbol{r}_k, \boldsymbol{r}_k). \tag{5.7}$$

Thus, we can compute $2m$ modified moments $\{\tilde{v}_k\}_{k=0}^{2m-1}$ during only $m + 1$ iterations, and these $\tilde{v}_k$ and the parameters $\{\tilde{\delta}_k\}_{k=0}^{2m-1}$ can be used as input to Algorithm 3.1 to determine a tridiagonal matrix T_m, from which we obtain a similar symmetric tridiagonal matrix $\hat{T}_m$ by (3.5).

We would like to evaluate only few inner products during the iterations with the adaptive Richardson iteration method. Therefore, we wish to compute modified moments only during a few iterations prior to each update of the interval $[a_j, b_j]$. This is possible because, as we will see now, for any nonnegative integer i, we can compute $2m$ modified moments $\{\tilde{v}_k\}_{k=2i}^{2m+2i-1}$ from the residual vectors $\{\boldsymbol{r}_k\}_{k=i}^{i+m}$, and use these moments and the parameters $\{\tilde{\delta}_k\}_{k=2i}^{2m+2i-1}$ to determine a tridiagonal matrix T_m by Algorithm 3.1. As before, we determine a similar symmetric matrix $\hat{T}_m$ from T_m by (3.5), and use the extreme eigenvalues and associated Gaussian weights of $\hat{T}_m$ to update the interval $[a_j, b_j]$ according to (5.3) and (5.4).

Let i be an arbitrary but fixed nonnegative integer. Define the polynomials

$$\tilde{p}_{k,i}(\lambda) := \prod_{j=2i}^{k+2i-1} (1 - \tilde{\delta}_j \lambda), \quad k = 1, 2, \ldots,$$

and $\tilde{p}_{0,i}(\lambda) := 1$. Introduce the modified moments associated with the polynomials $\tilde{p}_{k,i}$ and the measure $\mathrm{d}\sigma_{n,i} := \tilde{p}_{2i}(\lambda)\,\mathrm{d}\sigma_n$ by

$$\tilde{v}_{k,i} := \int_{-\infty}^{\infty} \tilde{p}_{k,i}(\lambda)\,\mathrm{d}\sigma_{n,i}, \quad k = 0, 1, \ldots.$$

Then

$$\tilde{v}_{k,i} = \tilde{v}_{k+2i}, \quad k = 0, 1, \ldots. \tag{5.8}$$

Thus, for an arbitrary value of i we can determine $2m$ modified moments $\{\tilde{v}_{k,i}\}_{k=0}^{2m-1}$ from (5.6)–(5.8), and use them together with the parameters $\{\tilde{\delta}_k\}_{k=2i}^{2m+2i-1}$, which define the polynomials $\{\tilde{p}_{k,i}\}_{k=0}^{2m-1}$, to determine a tridiagonal matrix T_m by the modified Chebyshev algorithm. We remark that the matrix $\hat{T}_m$ obtained by symmetrization of T_m can also be determined by the Lanczos process with initial vector $\boldsymbol{r}_i$.

5.4. Computation of modified moments

This subsection addresses the issue of when to compute modified moments. We would like to avoid determining modified moments very often, because their evaluation requires the computation of inner products, and this computation can be a bottleneck on some modern computers. We also note that the computation of the Euclidean vector norm requires the evaluation of an inner product. In order to keep the number of inner product computations small, we do not evaluate the norm of the residual error $\|\boldsymbol{r}_k\|$ in every iteration in Algorithm 5.2 below.

Assume that the relaxation parameters are generated by Algorithm 5.1 as reciprocal values of Leja points for the interval $[a_j, b_j]$. It follows from (1.3) that, for all $k \geqslant 0$,

$$\frac{\|\boldsymbol{r}_k\|}{\|\boldsymbol{r}_0\|} \leqslant \|p_k(A)\| = \max_{\lambda \in \lambda(A)} |p_k(\lambda)|.$$

Assume that for some $k > 0$ the inequality

$$\frac{\|\boldsymbol{r}_k\|}{\|\boldsymbol{r}_0\|} > \max_{\lambda \in [a_j, b_j]} |p_k(\lambda)| \tag{5.9}$$

holds. Then, clearly, $[a_j, b_j] \subsetneq [\lambda_1, \lambda_n]$, and it may be appropriate to increase the interval used to determine relaxation parameters. We remark that the maximum on the right-hand side of (5.9) is evaluated when determining Leja point z_k for $[a_j, b_j]$ by Algorithm 5.1. Note that the criterion (5.9) for updating the interval is independent of the expected asymptotic rate of convergence of the iterative method. Updating criteria that depend on the expected asymptotic rate of convergence, obtained by Lemma 2.1, are more difficult to use, because it is not easy to judge whether a finite number of iterates determined are indeed the first few elements of a sequence that converges with a desired rate.

Assume that the criterion (5.9) is violated for $k = i$. The above discussion suggests that it then may be appropriate to update the interval $[a_j, b_j]$ in the following manner: compute the modified moments $\{\tilde{v}_{j,i}\}_{j=0}^{2m-1}$ during the present and the next m iterations, use these modified moments to determine a symmetric tridiagonal matrix $\hat{T}_m$, and compute its eigenvalues and associated Gaussian weights. The latter quantities can then be used to determine a new interval $[a_{j+1}, b_{j+1}]$ as described in Sections 5.2 and 5.3.

Computational experience indicates that it is not worthwhile to try to update the intervals $[a_j, b_j]$ too frequently, because then we often obtain that $[a_{j+1}, b_{j+1}] = [a_j, b_j]$. In fact, $[a_j, b_j] \subsetneq [a_{j+1}, b_{j+1}]$ only if the residual vectors determined when using Leja points in the interval $[a_j, b_j]$ are sufficiently rich in eigenvector components associated with eigenvalues of A outside $[a_j, b_j]$. We try to achieve this by keeping each interval $[a_j, b_j]$, $j \geqslant 1$, for at least $2m$ iterations.

5.5. An algorithm for adaptive Richardson iteration

The computations with our adaptive Richardson method can be summarized as follows. An initial interval $[a_0, b_0]$, which may be a single point, has to be supplied. Its endpoints should satisfy $\lambda_1 \leqslant a_0 \leqslant b_0 \leqslant \lambda_n$. Then m Richardson iterations are carried out with relaxation parameters $\{\delta_k\}_{k=0}^{m-1}$ chosen to be reciprocal values of Leja points for $[a_0, b_0]$. If $a_0 = b_0$, then we let $\delta_k := 1/a_0$ for $0 \leqslant k < m$. During these iterations $2m + 1$ modified moments are evaluated, the first $2m$ of which are input to Algorithm 3.1. The value of the last modified moment, $\|\boldsymbol{r}_m\|^2$, is used to check whether $\|\boldsymbol{r}_m\|$ is sufficiently small to terminate the iterations. If this is not the case, then Algorithm 3.1 is used to determine the entries of an $m \times m$ tridiagonal matrix of the form (3.4). A diagonal similarity transformation of this matrix gives a symmetric tridiagonal matrix (3.5). The nodes and weights of the Gaussian quadrature rule (4.1) associated with the latter matrix are computed and used to update the interval $[a_0, b_0]$, i.e., $[a_0, b_0]$ is replaced by $[a_1, b_1]$ using formulas (5.3)–(5.4). Now m Richardson iterations are carried out with relaxation parameters $\{\delta_k\}_{k=m}^{2m-1}$ chosen to be reciprocal values of Leja points for $[a_1, b_1]$. When determining the relaxation parameters $\{\delta_k\}_{k=m}^{2m-1}$, the values of the previously used relaxation parameters $\{\delta_k\}_{k=0}^{m-1}$ are taken into account, as described by Algorithm 5.1.

If the rate of convergence of the iterates $\boldsymbol{x}_k$ generated is judged to be sufficiently rapid, i.e., if the inequality (5.9) is violated, then m more iterations are carried out before this inequality is checked again. On the other hand, if the inequality (5.9) is found to be true, then $2m + 1$ modified moments are evaluated during the following m iterations, and upon completion of these iterations Algorithm 3.1 and the similarity transformation (3.5) are used to determine an $m \times m$ symmetric tridiagonal matrix. Analogously as described above, the Gaussian quadrature rule associated with this matrix is computed and used to determine a new, possibly larger, interval $[a_2, b_2]$. The computations continue in this manner until the residual vector is sufficiently small. Details are provided by Algorithm 5.2.

This algorithm is used in the computed examples reported in Section 6. It is easy to see that only very few of the vectors used by the algorithm have to be stored simultaneously.

Algorithm 5.2 (Adaptive Richardson iteration)
Input: $A \in \mathbb{R}^{n\times n}$, $\boldsymbol{x}_0, \boldsymbol{b} \in \mathbb{R}^n$, $\varepsilon, a_0, b_0 \in \mathbb{R}$, $m \in \mathbb{N}$;
Output: *Approximate solution* $\boldsymbol{x}_k \in \mathbb{R}^n$;
$\boldsymbol{r}_0 := \boldsymbol{b} - A\boldsymbol{x}_0$;
$j := 0$; $k := 0$; $\varepsilon_w := 0.01/m$;
while $\|\boldsymbol{r}_k\|/\|\boldsymbol{r}_0\| \geqslant \varepsilon$ **do**
  if $\|\boldsymbol{r}_k\|/\|\boldsymbol{r}_0\| \leqslant \max_{a_j\leqslant\lambda\leqslant b_j} |p_k(\lambda)|$ **then**
    for $l := 1,2,\ldots,m$ **do**
      Compute δ_k *by Algorithm* 5.1 *for the interval* $[a_j, b_j]$;
      $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \delta_k \boldsymbol{r}_k$;
      $\boldsymbol{r}_{k+1} := \boldsymbol{b} - A\boldsymbol{x}_{k+1}$;
      $k := k + 1$;
    end l;
  else
    $\tilde{\nu}_{0,k} := \boldsymbol{r}_k^{\mathrm{T}} \boldsymbol{r}_k$;
    for $l := 1,2,\ldots,m$ **do**
      Compute δ_k *by Algorithm* 5.1 *for the interval* $[a_j, b_j]$;
      $\tilde{\delta}_{2k} := \delta_k$; $\tilde{\delta}_{2k+1} := \delta_k$;
      $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \delta_k \boldsymbol{r}_k$;
      $\boldsymbol{r}_{k+1} := \boldsymbol{b} - A\boldsymbol{x}_{k+1}$;
      $\tilde{\nu}_{2l-1,k-l+1} := \boldsymbol{r}_{k+1}^{\mathrm{T}} \boldsymbol{r}_k$;
      $\tilde{\nu}_{2l,k-l+1} := \boldsymbol{r}_{k+1}^{\mathrm{T}} \boldsymbol{r}_{k+1}$;
      $k := k + 1$;
    end l;
    if $\|\boldsymbol{r}_k\|/\|\boldsymbol{r}_0\| \geqslant \varepsilon$ **then**
      Use the modified moments $\{\tilde{\nu}_{l,k-m}\}_{l=0}^{2m-1}$ *and the parameters* $\{\tilde{\delta}_l\}_{l=2(k-m)}^{2k-1}$ *as input to Algorithm* 3.1 *to determine the tridiagonal matrix* $\hat{T}_m$ *of order* m;
      Symmetrize T_m *to obtain* $\hat{T}_m$ *according to* (3.5);
      Compute the Gaussian quadrature rule associated with $\hat{T}_m$ *by the Golub–Welsch algorithm with weights normalized according to* (4.3);
      Determine new interval endpoints a_{j+1} *and* b_{j+1} *by* (5.3) *and* (5.4);
      $j := j + 1$;
      for $l := 1,2,\ldots,m$ **do**
        Compute δ_k *by Algorithm* 5.1 *for the interval* $[a_j, b_j]$;
        $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \delta_k \boldsymbol{r}_k$;
        $\boldsymbol{r}_{k+1} := \boldsymbol{b} - A\boldsymbol{x}_{k+1}$;
        $k := k + 1$;
      end l;
    endif;
  endif;
endwhile;

It may be attractive to implement the for-loops in which x_{k+m} is computed from x_k and no modified moments are evaluated by the leapfrog variant described by Saylor [15]. Such an implementation requires fewer arithmetic operations with n-vectors than the for-loops in Algorithm 5.2, but requires factorization of polynomials of the form $q_{m-1}(z) = z^{-1}(1 - \prod_{j=k}^{k+m-1}(1 - \delta_j z))$.

6. Computed examples

In this section we present a few numerical examples that illustrate the performance of our adaptive Richardson scheme. The computer programs used were written in FORTRAN 77 and the numerical experiments were carried out on an IBM RISC 6000/550 workstation using double precision arithmetic, i.e., computations were carried out with approximately 15 significant decimal digits. The purpose of the numerical experiments is to show the convergence of the iterates generated by Algorithm 5.2, and to illustrate the computational work required. For matrices A of large order n, the computational effort is dominated by the evaluation of matrix–vector products and the computation of inner products between n-vectors; how the timings of these tasks compare depends on the sparsity of the matrix A as well as on the architecture of the computer used. We report for several examples the number of iterations required, which gives the number of matrix–vector products, and the number of inner product evaluations. We note that in order to measure these quantities we may restrict our attention to linear systems (1.1) with a diagonal matrix

$$A = \mathrm{diag}[a_{11}, a_{22}, \ldots, a_{nn}] \in \mathbb{R}^{n\times n}. \tag{6.1}$$

In all experiments we use the right-hand-side vector

$$\boldsymbol{b} := [a_{11}, a_{22}, \ldots, a_{nn}]^{\mathrm{T}}. \tag{6.2}$$

Results of the experiments are displayed in tables and figures. The latter show $\log_{10}(\|\boldsymbol{r}_k\|/\|\boldsymbol{r}_0\|)$ as a function of the iteration number k. We refer to the adaptive Richardson method (Algorithm 5.2) as AR in the tables and figures. The number of times Algorithm 5.2 determines a tridiagonal matrix $\hat{T}_m$ is shown in the tables in the column labeled "updates".

Algorithm 5.2 requires that an initial interval $[a_0, b_0]$ be specified. Choices of initial intervals are shown in the tables in the column labeled "initial interval". The final interval determined by Algorithm 5.2 is displayed in the column "final interval". The accuracy of the computed estimates of the extreme eigenvalues of the matrix A generally increases with the accuracy of the computed approximate solution. This is illustrated in Examples 6.2 and 6.3. We compare Algorithm 5.2 with the conjugate gradient CG method.

An iteration by Algorithm 5.2 requires fewer arithmetic operations with n-vectors than an iteration by the CG method. The exact number of vector operations required by an iteration in Algorithm 5.2 depends on whether modified moments are computed and whether the leapfrog implementation [15] is used. In order to illustrate the difference in arithmetic work required by Algorithm 5.2 and the CG method, we report the number of inner product evaluations of vectors in $\mathbb{R}^n$ required, where we also count each computation of the Euclidean norm of a vector in $\mathbb{R}^n$ as an inner product evaluation. The number of inner product evaluations is reported in the tables in the column "inner products". In all examples, we choose the initial approximate solution $\boldsymbol{x}_0 = \boldsymbol{0}$, and we let $m := 5$ in Algorithm 5.2.

Example 6.1. Let the linear system (1.1) be defined by (6.1) and (6.2) with $a_{ii} := i$ and $n := 4000$. Table 1 compares adaptive Richardson iteration for different initial intervals $[a_0, b_0]$ with the CG method when $\varepsilon = 1 \cdot 10^{-5}$. The table illustrates that the number of iterations required as well as the number of updates of the interval $[a_j, b_j]$ depends on the choice of initial interval. It is interesting to note that letting $[a_0, b_0]$ be the asymptotically optimal interval, i.e., $[a_0, b_0] = [1, 4000]$, does not yield the smallest number of iterations. The reason for this is that the residual polynomials $p_k(z)$ are small for z close to the smallest eigenvalue of A, even when the left endpoint a_j is larger than this eigenvalue. The very different initial intervals $[a_0, b_0] = [300, 3700]$ and $[a_0, b_0] = [3500, 3500]$ result in almost the same number of iterations. This indicates that the choice of initial interval is not critical for the performance of the adaptive Richardson method, as long as it is not too large. Not surprisingly, the interval used to determine Leja points has to be updated more often when the initial interval $[a_0, b_0]$ is small than when the initial interval is large. Table 1 shows Richardson iteration to be competitive with the CG method when the initial interval is not too large. Fig. 1 displays $\log_{10}(\|r_k\|/\|r_0\|)$ as a function of k. The continuous curve is for AR and the dashed curve for CG. The curves show $\log_{10}(\|r_k\|/\|r_0\|)$ for every k, but Algorithm 5.2 only computes $\|r_k\|$ every $m = 5$ iterations. In particular, the stopping criterion is checked only every m iterations by Algorithm 5.2. The residual vector r_{101} determined by Algorithm 5.2 satisfies $\log_{10}(\|r_{101}\|/\|r_0\|) < -5$ but the iterations continue until $\|r_{105}\|$ has been evaluated.

Example 6.2. Let the linear system (1.1) be defined by (6.1) and (6.2) with $a_{ii} := i^2$ and $n := 4000$. We define the initial interval by $a_0 := b_0 := (1/n)\,\mathrm{trace}(A) = 5.3 \cdot 10^6$. Table 2 compares the AR and CG methods for different values of ε. The AR method is competitive for all choices of ε. Note that the computed final interval $[a_j, b_j]$ is a better approximation of the asymptotically optimal interval $[1, 4000^2]$ when ε is smaller. Fig. 2 displays $\log_{10}(\|r_k\|/\|r_0\|)$ as a function of k for $\varepsilon = 1 \cdot 10^{-6}$. Note that $\log_{10}(\|r_k\|/\|r_0\|)$ increases monotonically from 0 to 1.1 as k increases from 0 to 5. When

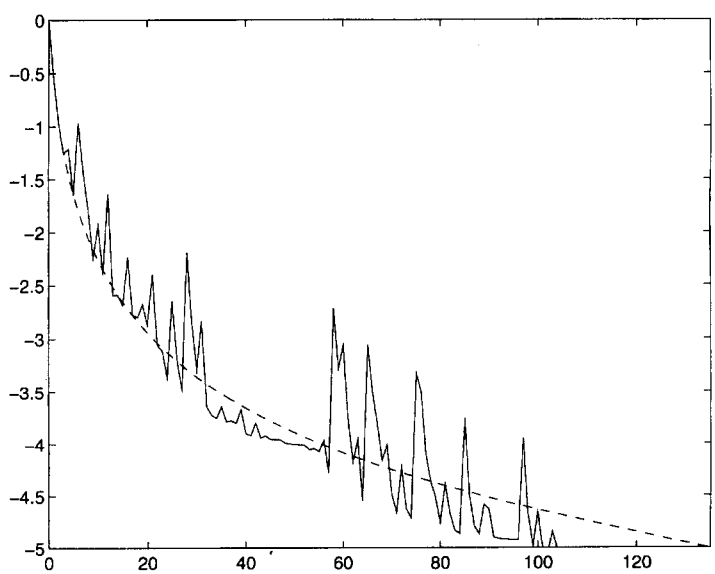

Fig. 1. $\log_{10}$ of relative residual error versus iteration number: – AR, –– CG.

Table 1
$A = \text{diag}[1, 2, \ldots, 4000]$, $\varepsilon = 1 \cdot 10^{-5}$

Iterative method	Updates	Iterations	Initial interval	Final interval	Inner products
AR	3	105	[300, 3700]	[11.64, 3998]	49
AR	5	110	[3500, 3500]	[4.212, 4000]	68
AR	0	190	[1, 4000]	[1, 4000]	39
CG		135			270

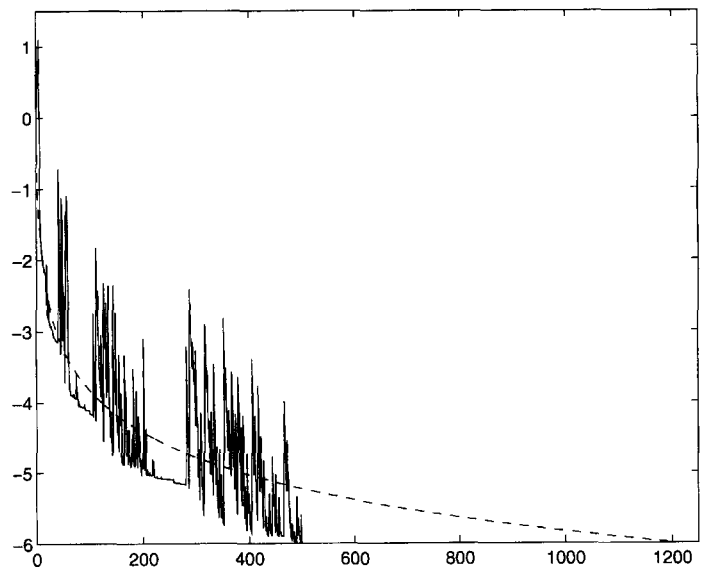

Fig. 2. $\log_{10}$ of relative residual error versus iteration number: – AR, – – CG.

Table 2
$A = \text{diag}[1^2, 2^2, \ldots, 4000^2]$, $[a_0, b_0] = [5.3 \cdot 10^6, 5.3 \cdot 10^6]$

Iterative method	ε	Updates	Iterations	Final interval	Inner products
AR	$1 \cdot 10^{-5}$	5	205	$[1.063 \cdot 10^4, 1.599 \cdot 10^7]$	87
CG	$1 \cdot 10^{-5}$		384		768
AR	$1 \cdot 10^{-6}$	7	500	$[1.711 \cdot 10^3, 1.600 \cdot 10^7]$	164
CG	$1 \cdot 10^{-6}$		1211		2422
AR	$1 \cdot 10^{-7}$	9	1305	$[2.327 \cdot 10^2, 1.600 \cdot 10^7]$	343
CG	$1 \cdot 10^{-7}$		3757		7514

$k = 5$, the initial interval is replaced by $[a_1, b_1] = [2.0 \cdot 10^6, 1.5 \cdot 10^7]$, and the residual error starts to decrease for $k > 5$. This example illustrates the importance of being able to update the interval $[a_0, b_0]$ after only few iterations.

Example 6.3. Let the matrix (6.1) of order $n := 4000$ be defined by $a_{ii} := (i/2000)^3$ and let the right-hand side vector be given by (6.2). In view of the fact that $(1/n)\text{trace}(A) \approx 1$, we let $a_0 := b_0 := 1$. Table 3 shows that the AR method requires much fewer iterations and inner

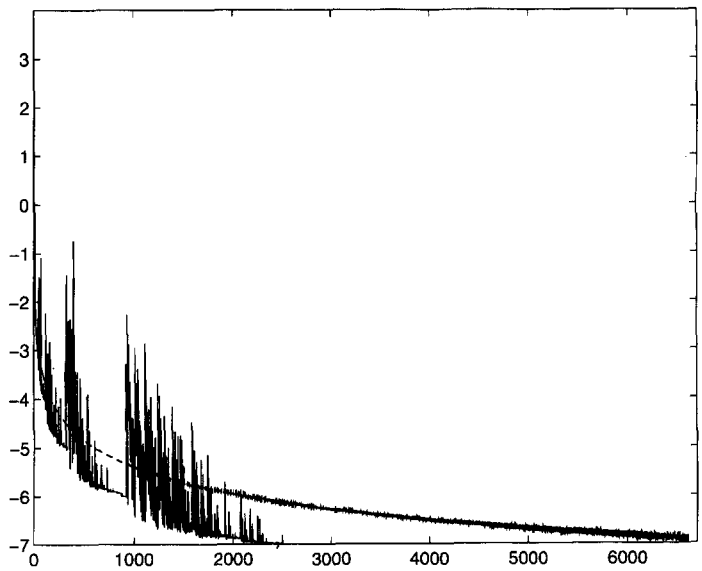

Fig. 3. $\log_{10}$ of relative residual error versus iteration number: — AR, -- CG.

Table 3
$A = \mathrm{diag}[(\frac{1}{2000})^3, (\frac{2}{2000})^3, \ldots, (\frac{4000}{2000})^3]$, $[a_0, b_0] = [1, 1]$

Iterative method	ε	Updates	Iterations	Final interval	Inner products
AR	$1 \cdot 10^{-5}$	6	380	$[6.307 \cdot 10^{-4}, 7.999]$	131
CG	$1 \cdot 10^{-5}$		607		1214
AR	$1 \cdot 10^{-7}$	8	2505	$[8.490 \cdot 10^{-5}, 8.000]$	574
CG	$1 \cdot 10^{-7}$		6616		13232

product evaluations than the CG method. Fig. 3 shows $\log_{10}(\|\boldsymbol{r}_k\|/\|\boldsymbol{r}_0\|)$ as a function of k for the AR and CG method when $\varepsilon = 1 \cdot 10^{-7}$. Similarly as in Example 6.2, the norm of the residual error increases during the first 5 iterations, i.e., while using the initial interval $[a_0, b_0]$ to generate relaxation parameters. The raggedness of the dashed curve for the CG method indicates numerical instability.

For the AR method, the interval $[a_j, b_j]$ is updated after $5, 20, 45, 75, 115, 315, 405$ and 920 iterations, and the left endpoint is moved after $5, 20, 45, 115, 315$ and 920 iterations. Fig. 4 displays $k \to \log_{10}(\|\boldsymbol{r}_k\|/\|\boldsymbol{r}_0\|)$ for the AR method for $\varepsilon = 1 \cdot 10^{-7}$, and shows that this curve oscillates most after each update of the left endpoint of the interval $[a_j, b_j]$. The amplitude of the oscillations is small when the interval has not been updated recently.

Presently, we do not know for which class of linear systems Algorithm 5.2 requires fewer iterations than the CG method. Comparing Algorithm 5.2 and the CG method for the solution of numerous linear systems suggests that the former almost always requires significantly fewer inner product evaluations. This makes Algorithm 5.2 attractive for implementation on parallel computers, on which the wall-clock time required for the evaluation of an inner product is nonnegligible compared with

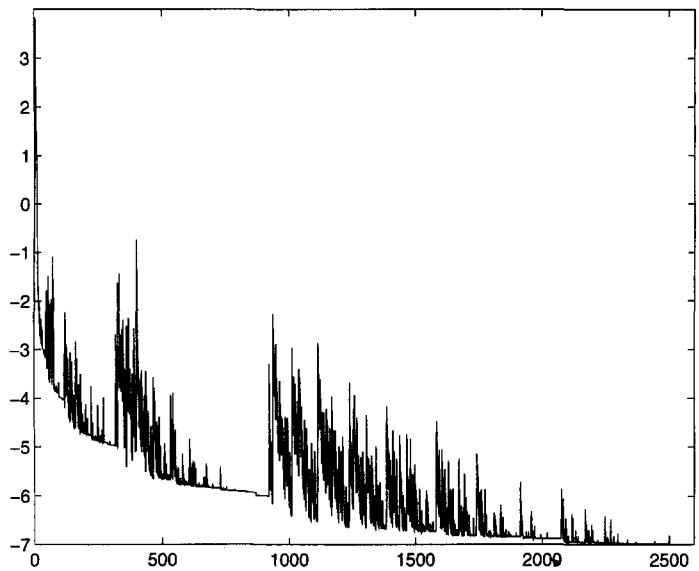

Fig. 4. $\log_{10}$ of relative residual error versus iteration number for AR.

the wall-clock time required for the computation of a matrix–vector product. Several massively parallel computers have this property.

7. Conclusion

We have demonstrated that an adaptive Richardson iteration method can require substantially fewer iterations than the CG method. The small number of inner product evaluations required by adaptive Richardson iteration makes this method attractive for implementation on parallel computers on which the evaluation of inner products is relatively slow due to the communication these evaluations require.

Acknowledgements

Work on this paper was carried out during visits to CERFACS and the University of Bologna. We would like to thank Mario Arioli, Iain Duff and Fiorella Sgallari for making these visits possible and enjoyable.

References

[1] D. Calvetti, G.H. Golub and L. Reichel, Gaussian quadrature applied to adaptive Chebyshev methods, in: G. Golub, A. Greenbaum and M. Luskin, Eds., *Recent Advances in Iterative Methods* (Springer, New York, 1994) 31–44.

[2] J.J. Dongarra, I.S. Duff, D.C. Sorensen and H.A. van der Vorst, *Solving Linear Systems on Vector and Shared Memory Computers* (SIAM, Philadelphia, PA, 1991).

[3] A. Edrei, Sur les déterminants récurrents et les singularités d'une fonction donnée par son developpement de Taylor, *Composito Math.* **7** (1939) 20–88.

[4] M. Eiermann, W. Niethammer and R.S. Varga, A study of semiiterative methods for nonsymmetric systems of linear equations, *Numer. Math.* **47** (1985) 505–533.
[5] W. Gautschi, On generating orthogonal polynomials, *SIAM J. Sci. Stat. Comput.* **3** (1982) 289–317.
[6] G.H. Golub, Some modified matrix eigenvalue problems, *SIAM Rev.* **15** (1974) 318–334.
[7] G.H. Golub and M.D. Kent, Estimates of eigenvalues for iterative methods, *Math. Comp.* **53** (1989) 619–626.
[8] G.H. Golub and C.F. Van Loan, *Matrix Computations* (Johns Hopkins Univ. Press, Baltimore, MD, 2nd ed., 1989).
[9] G.H. Golub and R.S. Varga, Chebyshev semi-iterative methods, successive over-relaxation methods, and second order Richardson iterative methods, *Numer. Math.* **3** (1961) 147–168.
[10] G.H. Golub and J.H. Welsch, Calculation of Gauss quadrature rules, *Math. Comp.* **23** (1969) 221–230.
[11] L.A. Hageman and D.M. Young, *Applied Iterative Methods* (Academic Press, New York, NY, 1981).
[12] K.C. Jea and D.M. Young, On the effectiveness of adaptive Chebyshev acceleration for solving linear systems, *J. Comput. Appl. Math.* **24** (1988) 33–54.
[13] F. Leja, Sur certaines suits liées aux ensembles plans et leur application à la representation conforme, *Ann. Polon. Math.* **4** (1957) 8–13.
[14] L. Reichel, The application of Leja points to Richardson iteration and polynomial preconditioning, *Linear Algebra Appl.* **154–156** (1991) 389–414.
[15] P.E. Saylor, Leapfrog variants of iterative methods for linear algebraic equations, *J. Comput. Appl. Math.* **24** (1988) 169–193.
[16] G. Szegő, *Orthogonal Polynomials* (Amer. Mathematical Soc., Providence, RI, 4th ed., 1975).
[17] R.S. Varga, *Matrix Iterative Analysis* (Prentice-Hall, Englewood Cliffs, NJ, 1962).